Compact mathematical formulation for Graph Partitioning

Marc BOULLE
France Telecom R&D
2, Avenue Pierre Marzin
22300 Lannion – France
marc.boulle@francetelecom.com

Abstract.
The graph partitioning problem consists of dividing the vertices of a graph into clusters, such that the weight of the edges crossing between clusters is minimized. We present a new compact mathematical formulation of this problem, based on the use of binary representation for the index of clusters assigned to vertices. This new formulation is almost minimal in terms of the number of variables and constraints and of the density of the constraint matrix. Its linear relaxation brings a very fast computational resolution, compared with the standard one.
Experiments were conducted on classical large benchmark graphs designed for comparing heuristic methods. On one hand, these experiments show that the new formulation is surprisingly less time efficient than expected on general k-partitioning problems. On the other hand, the new formulation applied on bisection problems allows to obtain the optimum solution for about ten instances, where only best upper bounds were previously known.

Key words. Linear programming, Graph Partitioning, Bisection

Introduction

The graph partitioning is a classical combinatorial optimization problem. This topic has several applications in the telecommunications field. For instance, the first step in the design of telecommunications network is to solve a partitioning problem. The network vertices are partitioned in order to minimize the amount of traffic between clusters of vertices. Furthermore, the graph partitioning is an appealing optimization problem, simple in its definition but NP-complete (Garey, Johnson and Stockmeyer 1976).
Partitioning a graph defined by its vertices, its edges and an edge cost function consists of dividing the vertices into clusters, such that the total weight of the edges whose endpoints are in different clusters is minimized. The number of clusters and their size are constrained. We consider the balanced partition of a graph, when the difference of size between clusters is at most one. The originality of our formulation is to use a binary representation for the index of clusters (assigned to vertices). The constraints in the standard partitioning formulation must be deeply reformulated to take advantage of this binary representation. The new formulation results into less number of variables, constraints and non null constraint coefficients. It is very efficient for bisection problems and allowed to obtain the optimal solution for several benchmark graphs not yet solved to optimality.
The remainder of the document is organized as follows:
Section one describes several new formulations and demonstrates their correctness. Section two evaluates the compactness of the formulations, and then the third section presents experimental results obtained by directly implementing the formulations in a mixed integer programming solver and applying them to classical large benchmark graphs.

1. Mathematical formulation

Let $G=(V, E)$ be a graph with vertex set V and edge set E. Let W_{ij} be the (positive) weight of the edge (i, j) between vertices i and j, K the maximum number of clusters k and $MaxCard$ the maximum size of each cluster ($|V| \leq K.MaxCard$). The objective is to minimize the cut, i.e. the total weight of the edges crossing the clusters. This is equivalent to maximizing the total weight of the edges that are inside the clusters.

1.1 Standard formulation

The standard straightforward integer programming formulation of the problem is the following.

Formulation A

Variables:

. v_{ik}=1 if vertex i belongs to cluster k and 0 otherwise

. e_{ijk}=1 if edge (i, j) belongs to cluster k and 0 otherwise

Maximize:

$$\sum_{i,j,k} e_{ijk} W_{ij}$$

Subject to:

Each vertex belongs to only one cluster *(1)*

$$\sum_{k} v_{ik} = 1 \qquad \forall i$$

Max size of clusters *(2)*

$$\sum_{i} v_{ik} \le MaxCard \qquad \forall k$$

Linearization of quadratic variables ($e_{ijk} = v_{ik} * v_{jk}$) *(3)*

$$e_{ijk} \le v_{ik}$$

$$e_{ijk} \le v_{jk}$$

$$e_{ijk} \ge v_{ik} + v_{jk} - 1 \qquad \forall i, j, k$$

Decision variables

v_{ik} binary variables

1.2 Slight compact formulation

The preceding formulation contains K variables and $3K$ constraints for each edge. This can be reduced to 1 variable and $2K$ constraints by introducing for each edge a new binary variable f_{ij} valued 1 if the edge (i,j) is inside one cluster and 0 otherwise. We show below that variables f_{ij} can be evaluated from existing vertex variables v_{ik}.

$$f_{ij} = \sum_{k} e_{ijk} = \sum_{k} v_{ik} v_{jk}$$

$$f_{ij} = 1 \Leftrightarrow \forall k / v_{ik} = v_{jk}$$

We can notice that $v_{ik} \neq v_{jk} \Leftrightarrow \left| v_{ik} - v_{jk} \right| = 1$. Thus, f_{ij} can be expressed in the following way:

$$f_{ij} = \underset{k}{Min}\left(1 - \left| v_{ik} - v_{jk} \right|\right)$$

As the problem of interest is a maximization problem in f_{ij} with positive weights, the preceding equality constraint can be turned into an inequality constraint, and rewritten in the following way.

$$f_{ij} \le \underset{k}{Min}\left(1 - \left| v_{ik} - v_{jk} \right|\right) \Leftrightarrow \forall k,\ f_{ij} \le 1 - \left| v_{ik} - v_{jk} \right|$$

This leads to the following formulation.

Formulation B

Variables:

. v_{ik}=1 if vertex i belongs to cluster k and 0 otherwise

. f_{ij}=1 if edge (i, j) belongs to one of the clusters and 0 otherwise

Maximize:

$$\sum_{i,j} f_{ij} W_{ij}$$

Subject to:

Each vertex belongs to only one cluster *(1)*

$$\sum_{k} v_{ik} = 1 \qquad \forall i$$

Max size of clusters *(2)*

$$\sum_{i} v_{ik} \le MaxCard \qquad \forall k$$

Constraints to calculate intra-cluster edge variables *(3)*

$$f_{ij} \le 1 + v_{ik} - v_{jk}$$

$$f_{ij} \le 1 - v_{ik} + v_{jk} \qquad \forall i, j, k$$

Decision variables

v_{ik} binary variables

1.3 Application to the bisection problem

In the case of bisection, one single decision variable is enough for each vertex. Its value is 1 if the vertex belongs to the first cluster and 0 otherwise. Formulation B can be simplified in the following way.

Formulation B2

Variables:

. v_{i1}=1 if vertex i belongs to first cluster and 0 otherwise

. f_{ij}=1 if edge (i, j) belongs to one of the clusters and 0 otherwise

Maximize:

$$\sum_{i,j} f_{ij} W_{ij}$$

Subject to:

Max size of clusters *(1)*

$$\sum_{i} v_{i1} \le MaxCard$$

$$\sum_{i} v_{i1} \ge |V| - MaxCard$$

Constraints to calculate intra-cluster edge variables *(2)*

$$f_{ij} \le 1 + v_{i1} - v_{j1}$$

$$f_{ij} \le 1 - v_{i1} + v_{j1} \qquad \forall i, j$$

Decision variables

v_{i1} binary variables

1.4 Formulation based on binary indexing of clusters

In the general case of k-partitioning, formulations A and B need K decision variables for each vertex. These variables theoretically enable 2^K choices for each vertex, although only K clusters are available. On this basis, we will aim at a more compact formulation, needing less variables and constraints. For each vertex i, we have to decide to which cluster it belongs. Let k_i be an integer variable corresponding to the index of the cluster assigned to vertex i ($0 \le k_i < K$). In its binary representation, variable k_i can be written as a linear combination of the powers of 2. The coefficients b_{ip} of this linear combination will be used as decision variables.

$$k_i = \sum_p b_{ip} 2^p \quad (0 \le p < \log2(K))$$

For example, if k_i=5, k_i=1*2^0+0*2^1+1*2^2, so b_{i0}=1=, b_{i1}=0, b_{i2}=1.

Starting from formulation B, we will rewrite its constraints with these new decision variables.

Each vertex belongs to only one cluster

This constraint is satisfied from the definition of k_i, independently of the choice of a decimal representation (variables k_i) or a binary representation (variables b_{ip}). Each vertex must be assigned to an existing cluster. This means that k_i (its binary representation) must be less than K-1.

Max size of clusters

The size of each cluster must be derived from the binary decision variables corresponding to the cluster indexes. Let us use again variables v_{ik} representing the assignment of vertex i to cluster k. We will show above how to calculate these variables from the new binary variables b_{ip}.

The binary representation of index k is based on constant values B_{kp}:

$$k = \sum_p B_{kp} 2^p$$

Variable v_{ik} equals 1 if and only if variable k_i equals k. We then have the following equivalences.

$$v_{ik} = 1 \Leftrightarrow k_i = k$$

$$v_{ik} = 1 \Leftrightarrow \forall p, b_{ip} = B_{kp}$$

$b_{ip} \setminus B_{kp}$	0	1
0	1	0
1	0	1

$$v_{ik} = 1 \Leftrightarrow \forall p, (1 - B_{kp})(1 - b_{ip}) + B_{kp} b_{ip} = 1$$

$$v_{ik} = 1 \Leftrightarrow \prod_p \left(1 - b_{ip} - B_{kp} + 2b_{ip}B_{kp}\right) = 1$$

So v_{ik} is a product of factors a_{ip}.

$a_{ip} = 1 - b_{ip} - B_{kp} + 2b_{ip}B_{kp}$ (if $B_{kp}=0$, $a_{ip}=1-b_{ip}$, otherwise $a_{ip}=b_{ip}$)

Variable v_{ik} can be evaluated with a generalization of the linearization of quadratic variables.

$$\forall p, v_{ik} \le a_{ip}$$

$$v_{ik} \ge 1 + \sum_p \left(a_{ip} - 1\right)$$

Thus, the avaibility of variables v_{ik}, derived from variables b_{ip}, allows to reuse the same formula for the constraints concerning the max size of clusters.

Constraints to calculate intra-cluster edge variables

Intra cluster edge variables equal 1 if the endpoints of the edge are in the same cluster, i.e. if the indexes of the clusters containing the endpoints of the edge have the same binary representation.

$$f_{ij} = 1 \Leftrightarrow \forall p / b_{ip} = b_{jp}$$

$$f_{ij} = \underset{p}{Min}\left(1 - \left|b_{ip} - b_{jp}\right|\right)$$

As the problem of interest is a maximization problem in f_{ij} with positive weights, the preceding equality constraint can be turned into an inequality constraint, and rewritten in the following way.

$$f_{ij} \le \underset{p}{Min}\left(1 - \left|b_{ip} - b_{jp}\right|\right) \Leftrightarrow \forall p,\ f_{ij} \le 1 - \left|b_{ip} - b_{jp}\right|$$

Formulation C

Let B_{kp} be the coefficient of 2^p in the binary representation of index k (B_{kp} is a constant).

Variables:

. b_{ip}: coefficient of 2^p in the binary representation of cluster index k_i assigned to vertex i

. f_{ij}=1 if edge (i, j) belongs to one of the clusters and 0 otherwise

. v_{ik}=1 if vertex i belongs to cluster k and 0 otherwise

Maximize:

$$\sum_{i,j} f_{ij} W_{ij}$$

Subject to:

Each vertex is assigned to a cluster index less than K *(1)*

$$\sum_p b_{ip} 2^p \le K - 1 \qquad \forall i$$

Max size of clusters *(2)*

$$\sum_i v_{ik} \le MaxCard \qquad \forall k$$

Constraints to calculate intra-cluster edge variables *(3)*

$$f_{ij} \le 1 + b_{ip} - b_{jp}$$

$$f_{ij} \le 1 - b_{ip} + b_{jp} \qquad \forall i, j, p$$

Linearization of variables used for vertices assignment to clusters *(4)*

$$v_{ik} \le \left(1 - b_{ip} - B_{kp} + 2b_{ip}B_{kp}\right) \qquad \forall i, k, p$$

$$v_{ik} \ge 1 + \sum_p \left(\left(1 - b_{ip} - B_{kp} + 2b_{ip}B_{kp}\right) - 1\right) \qquad \forall i, k$$

Decision variables

b_{ip} binary variables

1.5 Compact formulation based on binary indexing of clusters

Formulation C uses few decision variables, but increases the number of constraints to calculate the size of clusters. In order to decrease the number of variables and constraints used in formulation C, we show in this section how to calculate the size of clusters directly from the variables b_{ip} (binary representation of cluster indexes), without using previous variables v_{ik}.

First, let us define the notion of masking. An index *k* is masking a bit *p* if the binary representation of *k* has bit *p* set to 1. For example, the indexes 1 (1), 3 (11) and 5 (101) are masking bit 0.

Let us now evaluate the term $sb_0 = \sum_i b_{i0}$.

sb_0 stands for the number of vertices assigned to a cluster whose index is masking bit 0.

Thus, $sb_0 = \sum_i b_{i0} = \sum_{k / k \text{ is masking bit } 0} cardCluster_k = cardCluster_1 + cardCluster_3 + ...$

More generally, $sb_p = \sum_i b_{ip} = \sum_{k / k \text{ is masking bit } p} cardCluster_k$

The notion of masking can be generalized to bit masks. An index *k* is masking a bit mask *m* if the binary representation of *k* has all bits of the bit mask *m* set to 1. For example, with *m*=5 (101), the indexes 5 (101), 7 (111), 13 (1101) are masking the bit mask *m*.

Let bm_{im} be a binary variable, equal to 1 if cluster index k_i is masking bit mask *m*.

Let $sbm_m = \sum_i bm_{im}$.

$$sbm_m = \sum_{k / k \text{ is masking bit mask } m} cardCluster_k$$

For example, $sbm_5 = cardCluster_5 + cardCluster_7 + cardCluster_{13} + ...$

This leads to the following proposition.

Proposition 1: Variables sbm_m ($0 \le m < K$) can be linearly derived from the sizes of clusters $cardCluster_k$ ($0 \le k < K$).

Let SBM=M.C:

$$\begin{pmatrix} sbm_0 \\ sbm_1 \\ ... \\ sbm_{K-1} \end{pmatrix} = \begin{pmatrix} a_{00} & a_{01} & ... \\ a_{10} & a_{11} & \\ ... & & \\ a_{K-1.0} & a_{K-1.1} & \end{pmatrix} \begin{pmatrix} cardCluster_0 \\ cardCluster_1 \\ ... \\ cardCluster_{K-1} \end{pmatrix}$$

By definition, coefficient a_{mk} of matrix M equals 1 if the index *k* is masking the bit mask *m*.

Proposition 2: Matrix M is reversible.

Proof:

Each index *k* is masking the bit mask *k*, thus a_{kk}=1. If *k* is masking *m*, the binary representation of *k* is above the binary representation of *m* for each binary coefficient, and thus $k \ge m$. This can be summarized with the following properties:

$$a_{kk} = 1 \qquad \forall k$$
$$a_{mk} = 0 \qquad \forall m < k$$

As a conclusion, M is an upper diagonal matrix with all its diagonal terms set to 1. Hence, M is reversible.

Corollary 1: The sizes of clusters $cardCluster_k$ ($0 \le k < K$) can be linearly derived from the bit mask variables sbm_m ($0 \le m < K$).

Proposition 2 demonstrates that the size of clusters can be calculated from the new bit mask variables bm_{im}. We show in the following how to derive these bit mask variables bm_{im} from the decision variables variables b_{ip}.

First, in the special case of *m*=0, $sbm_0 = \sum_i bm_{i0} = \sum_k cardCluster_k = |V|$.

Second, we can notice that when the indexes of clusters are powers of 2, there is a correspondence between the two sets of variables: $bm_{im}=b_{ip}$ for $m=2^p$ ($0 \le p < \log(K)$).

Third, for $m > 0$ and $m \ne 2^p$, at least 2 bits in the binary representation of *m* are set to 1. Thus, $m=m_1+m_2$, where m_1 is a power of 2 (only one bit set to 1 in binary representation of m_1) and m_2 is strictly positive. Based on the definition of bit mask variables bm_{im}, we get $bm_{im} = bm_{im^1} bm_{im^2}$ and bm_{im^1} corresponds to one of the decision variables b_{ip}.

Therefore, the new bit mask variables bm_{im} can be calculated by recurrence from the decision variables b_{ip}. Using the linearization of quadratic variables, three constraints are necessary to calculate each new bit mask variable from two other variables (two decision variables, or one decision variable and one new smaller bit mask variable).

Let us illustrate these results with an example for K=6:

$$\begin{pmatrix} sbm_0 \\ sbm_1 \\ sbm_2 \\ sbm_3 \\ sbm_4 \\ sbm_5 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cardCluster_0 \\ cardCluster_1 \\ cardCluster_2 \\ cardCluster_3 \\ cardCluster_4 \\ cardCluster_5 \end{pmatrix}$$

- $sbm_0=|V|$
- sbm_1, sbm_2, sbm_4 are calculated with decision variables b_{i0}, b_{i1}, b_{i2}
- sbm_3 and sbm_5 are calculated with bit mask variables bm_{i3} and bm_{i5}
- $bm_{i3}=b_{i0}.b_{i1}$ (3=11=01+10 in binary representation)
- $bm_{i5}=b_{i0}.b_{i2}$ (5=101=001+100 in binary representation)

Formulation D

Let B_{kp} be the coefficient of 2^p in the binary representation of index k (B_{kp} is a constant).

Variables:

. b_{ip}: coefficient of 2^p in the binary representation of cluster index k_i assigned to vertex i
. bm_{im}=1 if the cluster index index is matching the bit mask m
. $cardCluster_k$: size of cluster k
. f_{ij}=1 if edge (i, j) belongs to one of the clusters and 0 otherwise

Maximize:

$$\sum_{i,j} f_{ij} W_{ij}$$

Subject to:

Each vertex is assigned to a cluster index less than K *(1)*

$$\sum_p b_{ip} 2^p \le K-1 \qquad \forall i$$

Equations to calculate the size of clusters *(2)*

$$\sum_k cardCluster_k = |V|$$

$$\sum_i b_{ip} = \sum_{k \,/\, k \text{ is masking bit } p} cardCluster_k \qquad \forall p$$

$$\sum_i bm_{im} = \sum_{k \,/\, k \text{ is masking bit mask } m} cardCluster_k \qquad \forall m \; (m \neq 2^p)$$

Max size of clusters *(3)*

$$cardCluster_k \le MaxCard \qquad \forall k$$

Constraints to calculate intra-cluster edge variables *(4)*

$$f_{ij} \le 1 + b_{ip} - b_{jp}$$

$$f_{ij} \le 1 - b_{ip} + b_{jp} \qquad \forall i, j, p$$

Linearization of quadratic variables bm_{im} ($bm_{im}=b_{ip}bm_{im'}$) *(5)*

$$bm_{im} \le b_{ip}$$

$$bm_{im} \le bm_{im'}$$

$$bm_{im} \ge b_{ip} + bm_{im'} - 1 \qquad \forall i, m \; (m \neq 2^p)$$

Decision variables

b_{ip} binary variables

2. Evaluation of the size of formulations

The sizes of formulations are evaluated in table 1 on the basis of the problem dimensions:

$|V|$: size of V
$|E|$: size of E
K: number of required clusters.

Formulation	Constraints	Variables	Binary variables	Non null coefficients
A (standard)	$\|V\| +$ $3\|E\|K +$ K	$\|V\|K +$ $\|E\|K$	$\|V\|K$	$2\|V\|K +$ $7\|E\|K$
B (A with fewer constraints)	$\|V\| +$ $2\|E\|K +$ K	$\|V\|K +$ $\|E\|$	$\|V\|K$	$2\|V\|K +$ $6\|E\|K$
B2 (B for K=2)	 $2\|E\| +$ K	$\|V\| +$ $\|E\|$	$\|V\|$	$\|V\|K +$ $6\|E\|$
C (binary representation)	$\|V\|(1+K+K\log(K)) +$ $2\|E\|\log(K) +$ K	$\|V\|(K+\log(K)) +$ $\|E\|$	$\|V\|\log(K)$	$\|V\|(\log(K)+2K+3K\log(K)) +$ $6\|E\|\log(K)$
D (compact binary representation)	$\|V\|(1+3(K-1-\log(K))) +$ $2\|E\|\log(K) +$ $2K$	$\|V\|(K-1) +$ $\|E\| +$ K	$\|V\|\log(K)$	$\|V\|(2\log(K) + 8(K-1-\log(K))) +$ $6\|E\|\log(K) +$ $K(K+1)$

Table 1. Size of formulations

Formulation B is slightly more compact than formulation A. In the case of bisection, formulation B2 and D are almost identical. In this case, using only one binary variable for cluster assignment is actually the same as using the binary representation of cluster indexes. Thus, formulation D can be seen as a generalization of formulation B2 for greater K. Formulation C is not interesting. This is only an intermediate step to introduce formulation D.

Comparison of the size of formulations B and D

The compactness of formulations B and D varies in opposite way for $|V|$ and $|E|$.

For the number of constraints: $\left|constr(D)\right| \le \left|constr(B)\right| \Leftrightarrow \frac{|E|}{|V|} \ge \frac{3}{2}\left(1-\frac{1}{K-\log(K)}\right)+\frac{1}{2|V|\left(1-\frac{\log(K)}{K}\right)}$

If $|E| \ge 3/2|V|$, the number of constraints in formulation D is smaller than in formulation B, and when $|E|/|V|$ gets larger, the reduction factor in the number of constraints becomes close from $\log(K)/K$.

The number of variables is similar in the two formulations. However, the number of binary variables is reduced with a factor $\log(K)/K$ in formulation D.

For the non null coefficients: $\left|coef(D)\right| \le \left|coef(B)\right| \Leftrightarrow \frac{|E|}{|V|} \ge 1-\frac{4}{3}\frac{1}{K-\log(K)}+\frac{K+1}{6|V|\left(1-\frac{\log(K)}{K}\right)}$

If $|E| \ge |V|$, the number of non null coefficients in formulation D is smaller than the number of non null coefficients in formulation B, and when $|E|/|V|$ gets larger, the reduction factor in the number of non null coefficients becomes close from $\log(K)/K$.

As a conclusion, formulation D is more compact than formulation B if $|E| \ge 3/2|V|$, and when $|E|/|V|$ gets larger, the compactness reduction factor becomes close from $\log(K)/K$ for the number of constraints, of non null coefficients and of binary variables.

Table 2 displays a numerical example for $|V|$=100, $|E|$=1000 and K=10. It shows a significant decrease in size in the new formulation for every dimension of the problem.

Formulation	Constraints	Variables	Binary variables	Non null coefficients
A	30110	11000	1000	72000
B	20110	2000	1000	62000
D	9620	1910	400	28910

Table 2: Size of formulations for $|V|$=100, $|E|$=1000 and K=10

Remark: The value of the linear relaxation of formulations A, B, B2, C and D is $\sum_{i,j} W_{ij}$, i.e. the sum of the weights of all edges of the graph.

3. Resolution with a solver

We compare the different formulations on the basis of numerical experiments by applying a solver on a benchmark.

3.1 Benchmark instances

The instances used for tests were introduced by (Johnson, Aragon, McGeoch and Schevon 1989). These instances fall into two categories: random graphs (presented in table 3) and random geometric graphs (presented in table 4). In random graphs, edges are randomly generated in order to reach an average vertex degree. In random geometric graphs, vertices are randomly located in square [0, 1]x[0, 1] and edges are created if their length is less than a distance d (d computed in order to reach a given average vertex degree). All weights of edges are set to 1.

|V| \ degree	2.5	5	10	20
124	G124.02	G124.04	G124.08	G124.16
250	G250.01	G250.02	G250.04	G250.08
500	G500.005	G500.01	G500.02	G500.04
1000	G1000.0025	G1000.005	G1000.01	G1000.02

Table 3. Random graphs

|V| \ degree	5	10	20	40
500	U500.05	U500.10	U500.20	U500.40
1000	U1000.05	U1000.10	U1000.20	U1000.40

Table 4. Random geometric graphs

3.2 Resolution environment

We used Cplex 6.5 library routines to solve the mixed integer formulation directly. For linear relaxation, we used the simplex dual method that proved much faster than the simplex primal method for all formulations. For exact resolution (mixed integer programming), we activated the rounding heuristic to set the initial value of binary variables, and the options of memory swapping of the branch and bound tree, not to be limited by the physical memory of the machine. The target machine is a PC Dell Workstation 400 with Pentium II 300 MHz and 256 MB RAM, operating under Windows/NT 4.0 system. This machine is rated 12.2 in SPECint95 benchmark.

3.3 Slightly stronger formulation

We can restrict the cluster indexes of the vertices in the following way:

- the first vertex must belong to first cluster
- the second vertex must belong either to first cluster, either to second cluster
- …
- the k^{th} vertex must belong to one of the k first clusters

These constraints can be added to each formulation:

Each vertex with index i<K is assigned to a cluster with index less than i

$$\sum_p b_{ip} 2^p \le i \qquad 0 \le i \le K-1$$

With these new constraints, the formulations are less degenerated because permutations of vertex assignment to clusters are now constrained. Experiments showed that the linear relaxations of formulations were improved with only a few percent compared with the previously established relaxation value. Exact resolutions were besides twice faster thanks to these new constraints.

3.4 Comparison of formulations in linear relaxation resolution

The value of the bound obtained with linear relaxation is not reported here. The only criterion used for comparison is CPU time in seconds. We used random graph G124.08 that have 124 vertices and an average vertex degree 10 (620 edges) for this comparison, whose results are displayed in table 5.

Formulation	3 clusters	4 clusters	5 clusters	6 clusters	8 clusters	16 clusters
A	8	19	26	42	105	604
B	6	17	39	74	190	1318
C	6	6	31	41	36	95
D	2	4	10	15	17	84

Table 5. CPU time for k-partitioning of instance G124.08 with linear relaxation

Formulation B is up to twice slower than formulation A, although it is slightly more compact. Formulation D, based on binary representation of cluster indexes, is much faster than standard formulation A, with a CPU time reduction factor larger than its compactness reduction factor log(*K*)/K

3.5 Comparison of formulations in exact resolution

Exact resolution is tractable with all formulations only with small size instances. We first evaluated all formulations on two 2-partitioning and one 3-partitioning problem. Results displayed in table 6 show that standard formulation A is clearly the slowest one. Formulation D, which is almost identical to formulation B2, is the fastest on bisection problems. Surprisingly, in spite of its compactness, formulation D based on the binary representation of cluster indexes is less efficient than formulation B on the 3-partitioning problem

Formulation	Graph G124.02 2 clusters	Graph G124.02 3 clusters	Graph U500.05 2 clusters
A	10	156	1273
B	2	137	196
B2	2	-	197
C	4	2220	198
D	2	366	184

Table 6. Exact resolution CPU time for 2 and 3-partioning of instances G124.02 and U500.05

In order to confirm that trend, we proceeded with other experiments of k-partitioning and compared formulations B and D on small random graphs with 30 to 50 vertices, tractable within reasonable CPU time. Table 7 reports the number of binary variables and the CPU time for formulations B and D, for each k-partitioning problem. In spite of its compactness, formulation D is always less time efficient than formulation B on general k-partitioning problems.
Tests with other solvers, using different resolution strategies, would be necessary to confirm these results.

Graph(|*V*|x|*E*|)	Clusters	Formulation B		Formulation D	
		Binary variables	CPU time	Binary variables	CPU time
G1(24x26)	6	144	9	72	344
G1(24x26)	7	168	18	72	1140
G1(24x26)	8	192	20	72	835
G2(28x66)	3	84	5	56	12
G2(28x66)	4	112	36	56	66
G3(47x60)	3	141	7	94	13
G3(47x60)	4	188	45	94	76

Table7. Exact resolution CPU time for several k-partitioning problems on three small random graphs

3.6 Optimum solution for large graphs

Despite the disappointing results for general k-partitioning problem, the resolution times obtained with bisection problems were promising. This was the reason for conducting new experiments to obtain as many as possible optimum solutions on benchmark graphs. These graphs are large size graphs which are used to evaluate and to compare heuristic solving methods. The purpose of these last experiments is not to evaluate the different formulations, but to find optimal solutions on benchmark graphs. That is why we chose what seemed to be the fastest formulation when used with our straightforward implementation in Cplex solver. For bisection problems, we used formulation B2, which is almost identical to formulation D in this case. For k-partitioning problems, we chose formulation B.

Bisection

The best known values come from (Johnson, Aragon, McGeoch, Schevon 1989) for small instances ($|V| \leq 250$) and from (Battitti and Bertossi 1999) for large instances ($|V| \geq 500$). These upper bounds correspond to the best values obtained by any heuristic method. Resolutions with formulation B2 were conducted on part of the instances. Some of these resolutions were performed until optimality was reached. Some other ones were stopped if optimality was untractable. The last ones were not conducted at all when smaller similar instances were not solvable.
The results are displayed in the tables 8 and 9. The optimal cuts found with exact resolution are in bold face, with the CPU time (seconds). For the other instances, the best known upper bounds found by heuristic methods are reported.

$\|V\|$ \ degree	2.5		5		10		20	
	Cut	CPU	Cut	CPU	Cut	CPU	Cut	CPU
124	**13**	2	**63**	2168	178		449	
250	**29**	228	114		357		828	
500	**49**	1245	218		626		1744	
1000	95		445		1362		3382	

Table 8. Results for bisection of random graphs with formulation B2

$\|V\|$ \ degree	5		10		20		40	
	Cut	CPU	Cut	CPU	Cut	CPU	Cut	CPU
500	**2**	195	**26**	970	**178**	373800	412	
1000	**1**	1479	**39**	3480	222		737	

Table 9. Results for bisection of random geometric graphs with formulation B2

Optimal cuts were obtained on 9 out of the 24 instances. These results obtained with formulation B2 prove the optimality of the corresponding upper bounds previously reached by heuristic methods. These new results (proven optimality on 9 instances) provide an absolute criterion to evaluate heuristics. It is surprising that such large instances could be solved to optimality. The reason is that the solved instances have a very low cut, and that their linear relaxation is in this case, very close to the optimum.
Computation time needed to prove optimality increases quickly with the size of the graphs and the value of the cut. Random geometric graphs are easier to solve than random graphs with the integer programming formulation. These graphs are more structured and thus the solution space is less degenerated than in the case of random graphs. This leads to a quicker exploration of the branch and bound tree during the resolution.

K-partitioning

Optimal values were obtained only with two small random graphs, using formulation B. Results are reported in tables 10 and 11

Graph G124.02	Optimal cut	CPU time
2 clusters	13	2
3 clusters	18	150
4 clusters	23	3840
5 clusters	25	133740

Table10. Results for k-partitioning of instance G124.02 with formulation B

Graph G250.01	Optimal cut	CPU time
2 clusters	29	228
3 clusters	41	241200

Table 11. Results for k-partitioning of instance G250.01 with formulation B

These results provide a benchmark for the k-partitioning problem. Computation time increases extremely quickly with the number of clusters. We used formulation B because it was more efficient than formulation D with Cplex solver. Formulation D still remains very promising because of its reduced size (particularly its reduced number of binary variables). We expect that using formulation D with other solvers or with a specially designed resolution method could improve significantly the efficiency of the resolution, and therefore bring new optimum values.

3.7 Quality of the state of the art heuristics

The graph bisection problem has been extensively studied in the past, and many heuristics have been experimented. For example, algorithms such as (Kernighan-Lin 1970) or (Fiduccia-Mattheyes 1982) are frequently used to locally improve bisections. Many meta-heuristics have also been used such as simulated annealing (Kirkpatrick, Gellat and Vecchi 1983) evaluated by (Johnson, Aragon, McGeoch and Schevon 1989), genetic algorithms used by (Bui and Moon 1996) or tabu search (Glover 1989) enhanced and adapted to bisection problem by (Battiti and Bertossi 1999). The multilevel approach is specially fitted to very large graphs and constrained computation time. It has been presented and studied by (Hendrickson and Leland 1995), (Monien and Diekmann 1997), (Pellegrini and Roman 1996), (Karypis and Kumar 1998). These families of heuristics represent a range of options for the trade-off between computation time and quality of the solution. In the light of the 9 optimal solutions proven in this paper, we shortly review the results of some of the previously presented heuristics, from the point of view of the quality of solutions. The multilevel approach that favors extremely short computation time at the expense of the quality of solutions is not evaluated.
The first evaluation deals with the results of (Johnson, Aragon, McGeoch and Schevon 1989), that introduced the benchmark instances studied in this paper and extensively studied heuristics based on simulated annealing. Table 12 compares optimal values with best values found by the authors in 1989 with their simulated annealing based method. These early works allowed to reach 6 of the 9 optimum cut sizes. The 3 non optimal values are indicated with stars in table 12.

Graph	Optimal cut	Best Simulated Annealing cut size
G124.2.5	13	13
G124.05	63	63
G250.2.5	29	29
G500.2.5	49	*52
U500.05	2	*4
U500.10	26	26
U500.20	178	178
U1000.05	1	*3
U1000.10	39	39

Table 12. Optimal versus best values for annealing method

The second evaluation is based on results presented by Bui and Moon 1996 for their method based on genetic algorithms (BFS-GBA), and Battiti and Bertossi 1999 for their method based on tabu search (RRTS). The instances with less than 500 nodes are now considered as too easy and are not taken into account by the community. Computation times have been scales with respect to SpecINT95 to be comparable. We compare the minimal and average cuts of 1000 heuristic runs with the optimal cut. This evaluation shows that state of the art heuristics have reached the optimal cut sizes for these 6 instances. The average cut size of RRTS method is optimal for all instances, except for G500.2.5. Computation time needed to prove the optimal solution with the exact resolution method is about 1000 times that of heuristic methods.

Graph	Optimal resolution		BFS-GBA			RRTS, 1000 n iter		
	Optimal	CPU	Min	Average	CPU	Min	Average	CPU
G500.2.5	49	1245	49	53,97	0,22	*51	52,06	0,98
U500.05	2	195	2	3,65	0,29	2	2	0,83
U500.10	26	970	26	32,68	0,37	26	26	1,32
U500.20	178	373800	178	179,58	0,44	178	178	2,59
U1000.05	1	1479	1	1,78	0,67	1	1	2,05
U1000.10	39	3480	39	55,78	1,19	39	39,03	3,08

Table 13: Optimal versus best known values for two state of the art methods based on genetic algorithms and tabu search

Conclusion

The new formulation of k-partitioning problem is more compact with an average reduction factor of $\log(K)/K$ compared with the standard formulation. In particular, the number of binary variables used in the new formulation is exactly reduced by a factor $\log(K)/K$. This compact formulation is promising and could be the basis for efficient exact

resolution methods or bounding methods. A straightforward use of a commercial solver was conclusive for bisection problems, but resulted into an increased computation time for the resolution of general k-partitioning problems. This new formulation is interesting, but further work will be necessary to conduct experiments with other solvers or to design a dedicated exact resolution method that could take advantage of the compactness of the formulation.
Experiments were conducted on some large size benchmark graphs, classically used to compare heuristic methods. For all these instances, previous work reported best known values, i.e. upper bounds. These experiments, based on the new formulation for bisection and on an almost standard formulation for general k-partitioning problems, allowed to obtain more than 10 optimum values on instances up to 1000 vertices and 5000 edges. These results provide a strong evaluation criteria to compare heuristic methods.

References

R. Battiti and A.A. Bertossi, 1999. “Greedy, Prohibition, and Reactive Heuristics for Graph Partitioning”, IEEE Transactions on Computers, vol. 48, no. 4. pp. 361-385.

T. N. Bui and B. R. Moon, 1996. “Genetic Algorithm and Graph Partitioning”, IEEE Transactions on Computers, vol. 45, no. 7. pp. 841-855.

C. Fiduccia and R. Mattheyses, 1982. “A Linear Time Heuristic for Improving Network Partitions”, Proc. 19th ACM/IEEE Design Automation Conf., Las Vegas, pp. 175-181.

M.R Garey, D.S. Johnson and L. Stockmeyer, 1976. “Some simplified NP-complete graph problems”, Theoretical Computer Science, 1, pp. 237-267.

F. Glover, 1989. “Tabu Search – Part I”, ORSA I Computing, vol1, no3, pp. 190-260.

B. Hendrickson and R. Leland, 1995. “A multilevel algorithm for partitioning graphs”, Proc. Supercomputing ’95, ACM.

D.S. Johnson, C.R. Aragon, L.A. McGeoch and C. Schevon, 1989. “Optimization by Simulated Annealing: An Experimental Study; Part 1, Graph Partitioning”, Operations Research, vol. 37, pp. 865-892.

G. Karypis and V. Kumar. (1998). “A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs”, SIAM J. on Scientific Computing, to appear.

B. Kernighan and S. Lin, 1970. “An Efficient Heuristic Procedure for Partitioning Graphs”, Bell Systems Technical J., vol. 49, pp. 291-307.

S. Kirkpatrick, C.D. Gellat Jr. and M.P. Vecchi, 1983. “Optimization by Simulated Annealing”, Science, vol. 220, no. 4598, pp.671-680.

B. Monien and R. Diekmann, 1997. “A Local Graph Partitioning Heuristic Meeting Bisection Bounds”, Proc. Eighth SIAM Conf. Parallel Processing for Scientific Computing.

F. Pellegrini and J. Roman, 1996. “Scotch: A Software Package for Static Mapping by Dual Recursive Bipartitioning of Process and Architecture Graphs”, Proc. HPCN’96 Brussels, pp. 493-498.